

# DevOps

## Table of Contents

Introduction .....	1
Nexus .....	1
1. Introduction .....	1
2. Nexus Repository OSS .....	2
2.1. Installation .....	2
2.1.1. Windows .....	2
2.2. Accessing the User Interface .....	3
2.3. Deploying Packages to NuGet Hosted Repositories .....	3
2.3.1. Accessing the Api Key .....	3
2.3.2. Setting the Api Key .....	4
2.3.3. Deploying the packages using nuget.exe tool .....	4
Package Management Systems .....	4
Introduction .....	5
NuGet .....	5
Introduction .....	5
1. Command line tools .....	5
2. Package management .....	5
2.1. Configuration .....	5
2.2. Package information .....	6
2.3. What happens when a NuGet package is installed .....	6

## Introduction

## Nexus

Doc Writer <[christian.popescu@outlook.com](mailto:christian.popescu@outlook.com)> v 1.0, 2020-06-24 :sectnums: :toc: :toclevels: 5

## 1. Introduction

This article is about Nexus. It manages binaries and build artifacts across your software supply chain.

There are two versions

¥ Commercial - Nexus Repository Pro

## 2. Nexus Repository OSS

The free artifact repository with universal format support.

[Official Site](#)

### 2.1. Installation

[Installation Methods](#)

#### 2.1.1. Windows

¥ Download package

¥ Unzip package to a choosen folder Example: F:\Applications\nexus-3.24.0-02-win64

¥ Tree structure

Example:

```
/f/Applications/nexus-3.24.0-02-win64
$ tree -L 2
.
|-- nexus-3.24.0-02
|   |-- NOTICE.txt
|   |-- OSS-LICENSE.txt
|   |-- PRO-LICENSE.txt
|   |-- bin
|   |-- deploy
|   |-- etc
|   |-- jre
|   |-- lib
|   |-- public
|   `-- system
`-- sonatype-work
```

¥ The Nexus Repository Manager executable nexus.exe can be found inside the bin directory

¥ It could be run with the following command. When run like this Nexus is started in shell window.

```
nexus.exe /run
```

¥ The application can be accessed once the log shows the message "Started Sonatype Nexus". The running application can be stopped using CTRL+C at the appropriate console.

¥ The nexus.exe executable can be used to manage the repository manager as a service with the /start, /stop, /restart, /force-reload and /status commands.

## 2.2. Accessing the User Interface

¥ Once the repository manager is started, the application is listening on the configured IP address range and port. By default any IP address and port 8081 are used.

URL `http://<server_host>:<port>` e.g. `http://localhost:8081/`

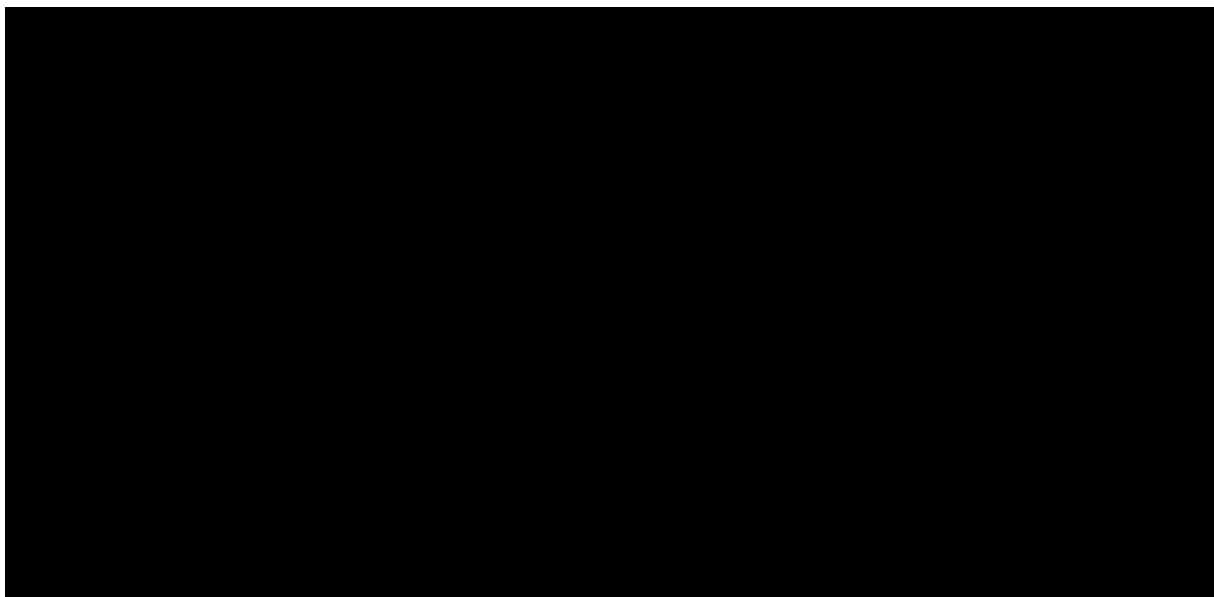
¥ The repository manager installation includes an administrative user with full access. Its username is admin and the initial password can be found in an admin.password file in the \$data-dir directory. You can sign in with the button on the top right corner of the user interface.

## 2.3. Deploying Packages to NuGet Hosted Repositories

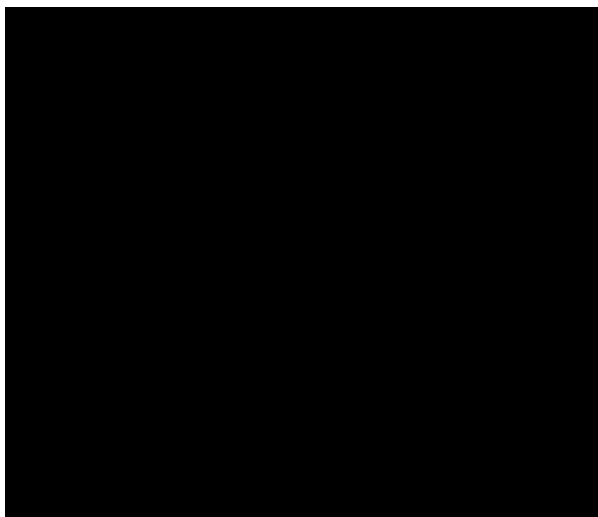
In order to authenticate a client against a NuGet repository, NuGet uses an API key for deployment requests. The API key acts as an alias for the user account, so the same API key is used for all NuGet repositories within the repository manager. This user-specific key is generated separately by a user and can be regenerated at any time. At regeneration, all previous keys generated for that user are invalidated.

### 2.3.1. Accessing the Api Key

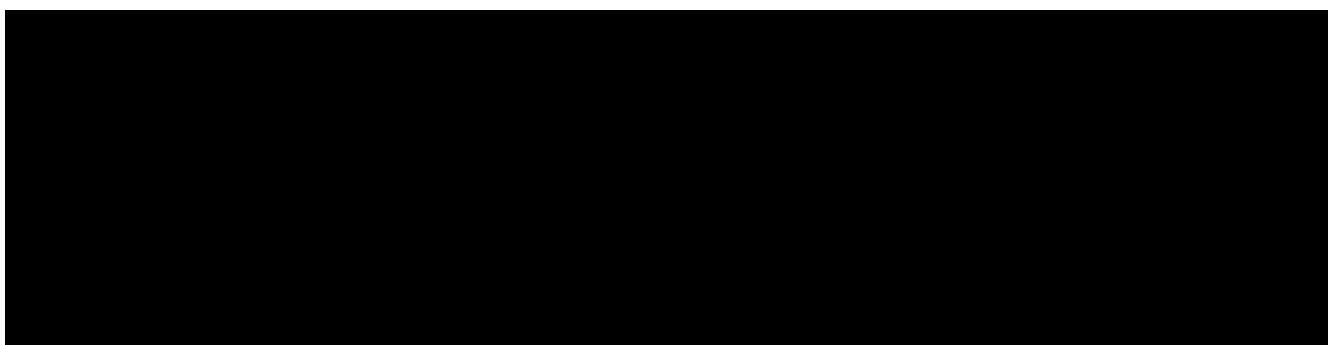
For usage with the repository manager, NuGet API keys are only needed when packages are going to be deployed. Users with the necessary apikey-all security privilege can access the NuGet API Key feature view via the User menu by pressing on their username in the main toolbar.



Select button



Dialog is shown



Usage of the API key requires the NuGet API-Key Realm to be activated. To do this, simply add the realm to the active realms in the Realms feature of the Security menu from the Administration menu.

### 2.3.2. Setting the Api Key

Setting the Api Key for a given repository using nuget.exe tool example:

```
nuget setapikey a1166690-be7f-31ba-8ce9-85aae086ac01 -source  
http://localhost:8081/repository/sandbox
```

### 2.3.3. Deploying the packages using nuget.exe tool

Example:

```
nuget push DSALibrary.1.0.1.nupkg -source  
http://localhost:8081/repository/Sandbox_new/
```

# Package Management Systems

Doc Writer <[christian.popescu@outlook.com](mailto:christian.popescu@outlook.com)> v 1.0, 2020-06-24

# Introduction

## NuGet

Doc Writer <[christian.popescu@outlook.com](mailto:christian.popescu@outlook.com)> v 1.0, 2020-06-24

## Introduction

NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages.

### 1. Command line tools

¥ dotnet.exe CLI

The .NET Core CLI, dotnet.exe, works on all platforms (Windows, Mac, and Linux) and provides core NuGet features such as installing, restoring, and publishing packages. dotnet provides direct integration with .NET Core project files (such as .csproj), which is helpful in most scenarios. dotnet is also built directly for each platform and does not require you to install Mono.

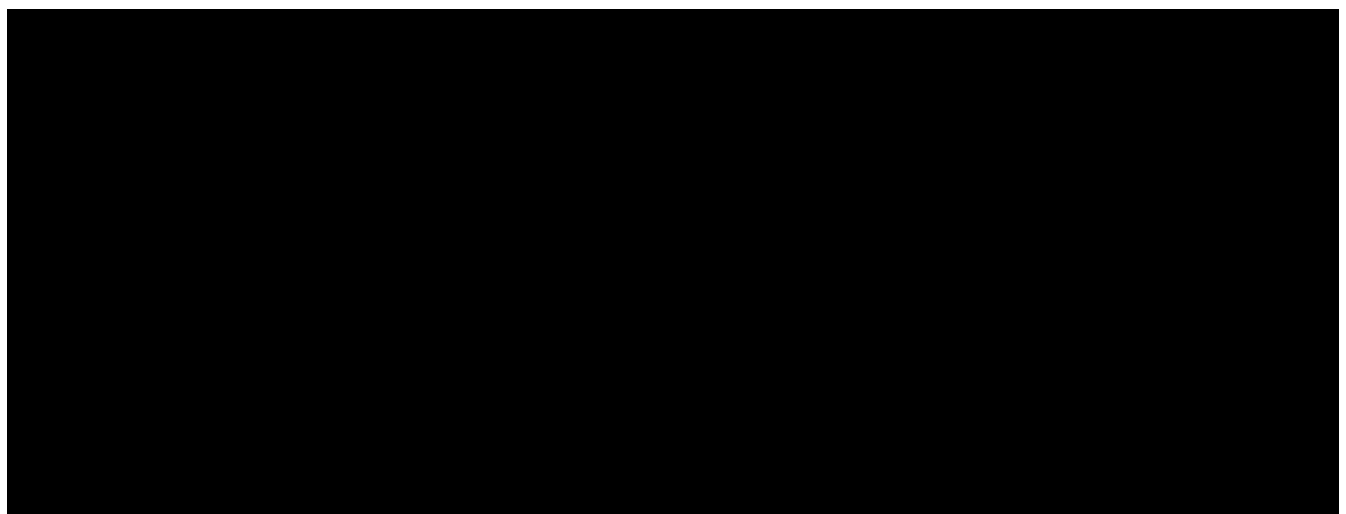
¥ nuget.exe CLI

The nuget.exe CLI, nuget.exe, is the command-line utility for Windows that provides all NuGet capabilities.

### 2. Package management

#### 2.1. Configuration

User configuration file location and content example



## 2.2. Package information

¥ List packages for a given source

```
nuget list -Source http://localhost:8081/repository/Sandbox_new
```

¥ Install package to a given project (packages folder)

```
nuget install <packageID> -OutputDirectory packages
```

¥ Remove package

To delete one or more packages, delete the packages you want to remove from the packages folder.

If you want to reinstall packages, use the *restore* or *install* command.

¥ Viewing folder locations

```
# Display locals for all folders: global-packages, http cache, temp and plugins  
cache  
nuget locals all -list
```

Typicaly output:

```
http-cache: C:\Users\user1\AppData\Local\NuGet\v3-cache  
global-packages: C:\Users\user1\.nuget\packages\  
temp: C:\Users\user1\AppData\Local\Temp\NuGetScratch  
plugins-cache: C:\Users\user1\AppData\Local\NuGet\plugins-cache
```

## 2.3. What happens when a NuGet package is installed

the different NuGet tools typically create a reference to a package in the project file or packages.config, then perform a package restore, which effectively installs the package. The exception is nuget install, which only expands the package into a packages folder and does not modify any other files.

The general process is as follows:

1. (All tools except nuget.exe) Record the package identifier and version into the project file or packages.config.
2. Acquire the package
3. Save a copy of the package and other information in the *http-cache* folder.
4. If downloaded, install the package into the per-user global-packages folder.

5. NuGet installs package dependencies as required.
6. Update other project files and folders:
  - a. For projects using PackageReference, update the package dependency graph stored in obj/project.assets.json. Package contents themselves are not copied into any project folder.
  - b. Update app.config and/or web.config if the package uses source and config file transformations.
7. (Visual Studio only) Display the package's readme file, if available, in a Visual Studio window.

Details on : [Package Installation Process](#)